Evaluation of Machine Learning for Pneumonia Classification

CS 445/545

Fall Term, 2019

David Poole, Andre Mukhsia

1. Introduction

1.1 Background

For Machine learning in recent years there has been an astounding success in the field of medical diagnosis. Computer vision has proven successful in many research studies suggesting that advances in the field may eventually lead to a world where the doctor's eyes are less capable then the Machine Learning algorithms for diagnosing patients for a wide variety of illness. In this experiment we will be examining the potential to apply two different machine learning algorithms to the task of diagnosing pneumonia.

In this report we will be comparing the efficacy of the Convolutional Neural Network (CNN) algorithm and the Random Decision Forest (RDF) algorithm for the problem of medical diagnosis for pneumonia using supervised learning. This report will go over the methods, dataset and algorithms used to accomplish our task and will include a detailed comparison of the two algorithms and how they were used in this experiment.

1.2 Hypothesis

Through the usage of a dataset comprised of X-ray image scans of lungs sorted by normal and pneumonia diagnosed lungs, we should be able to apply Machine Learning algorithms to fulfill the proposition that: Using a Convolutional Neural Network or a Random Forest Classifier it should be possible to predict within a reasonable scale (Accuracy> 85%) whether there is pneumonia symptoms in a given Chest X-ray image.

2. Methods

2.1 Algorithms

2.1.1 Algorithms Overview

The algorithms used in this experiment are the Convolutional Neural Network (CNN) and the Random Decision Forest classifier (RDF). The CNN algorithm is a class of deep neural network that has proven incredible results in analyzing images. The CNN was a fantastic choice for an algorithm for the given task due to its ability to analyze images and classify data in the form that we are given in this experiment. The RDF algorithm is an ensemble learning method that has been used successfully as a capable classification and regression model. The RDF algorithm has not been proven to have the same level of efficacy in usage with image data analysis which makes it a great algorithm to compare with the abilities of the Convolutional Neural Network.

2.1.2 Convolutional Neural Network

The Convolutional Neural network is an algorithm similar in structure to a perceptron or neural network architecture with the added complexity of a reguralized multilayer perceptron. A CNN is designed with multiple subsequent layers of perceptrons each with there own task and purpose(Karpathy). The layers of the CNN are split into separate layers with individual paradigms such as the Convolutional, Pooling, Fully Connected, Max Pooling, Dropout, Normalization, Flatten, and Feature Maps. The hidden layers of the CNN are used to prevent overfitting and can be seen as a corollary to the biological layers that connect the receptive fields of animal visual cortex(Hubel).

2.1.3 Random Decision Forest

A Random Decision Forest is made up of multiple decision trees. Its basic unit, the decision tree, works by using observed features at each tree nodes to split data into groups which are as different from each other as possible, resulting into subgroups with similar members which should be the correct grouping of data to the corresponding class labels. Given a data to predict, its features will be checked at each tree nodes until a leaf is reached, giving back a class prediction. Each trees in a Random Decision Forest outputs a class prediction vote where the class with the majority of the votes wins, which will work better than an individual Decision Tree; A large number of uncorrelated trees protect each other from individual errors (Yiu, 2019). Training involves creating a forest of many decision trees, each given a random set of data from the pool of training data and choosing random number of features to observe.

2.1.4 Algorithms Compared

As both of the algorithms used in this experiment fundamentally are classifiers they both fit for the intended purpose of addressing the problem of pneumonia diagnosis. The differences in the two algorithms lead well to comparison as they structurally are very different in the way they classify input data. Random Forest algorithms are often used in the banking and ecommerce sector as they work well with very large datasets and handle outliers and overfitting. The dataset used with images generates a lot of managed data and it may do well with this intended task. At the beginning of this experiment we had made the hypothesis that it is most likely that the Convolutional Neural Network will perform better in classification as far as accuracy percentages are concerned. The CNN algorithm is the go to algorithm for any Image analysis task and it will most likely have a much higher accuracy then the RDF algorithm will for this particular dataset.

2.2 Data Set

2.2.1 Data Set Overview

The dataset used for this experiment was posted to kaggle in 2017 by Paul Mooney and can be found: here. The dataset is comprised of 5863 X-ray images in three directories: (1). Train, (2) test and (3) val, these directories are the training , testing and validation set respectively. Each directory has two subsections for healthy lung images labeled NORMAL (fig

2.A.2.1) and for pneumonia diagnosed lung images labeled PNEUMONIA (fig 2.A.2.2). The chest X-ray images in this dataset were selected from cohorts of a pediatric study of patients at the Guangzhou Women and Children's Medical Center. The X-ray images all come from patients age one to five and all images have been screened for quality and readability. All diagnoses in the dataset and organization in two the two subsections were graded and diagnosed by two expert physicians and then cleared by a legacy AI system, the validation set was additionally screened for correctness by a third physician.

Fig 2.A.2.1 (NORMAL X-ray example)

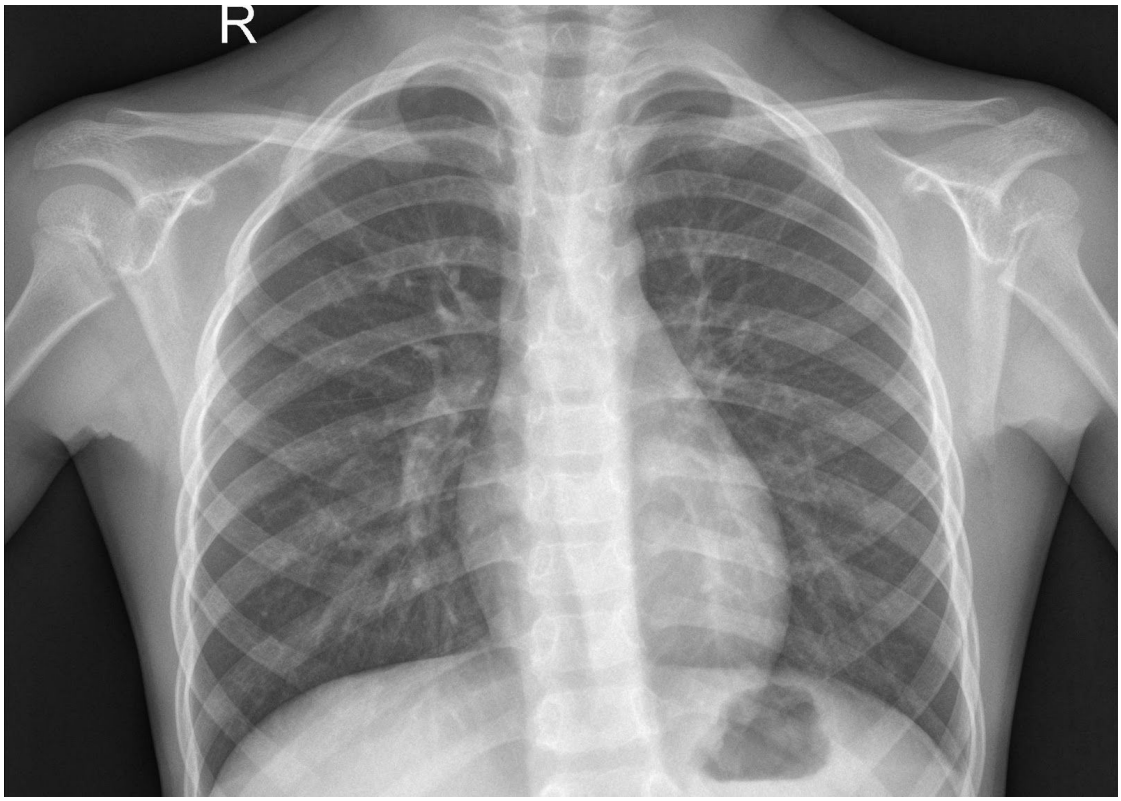
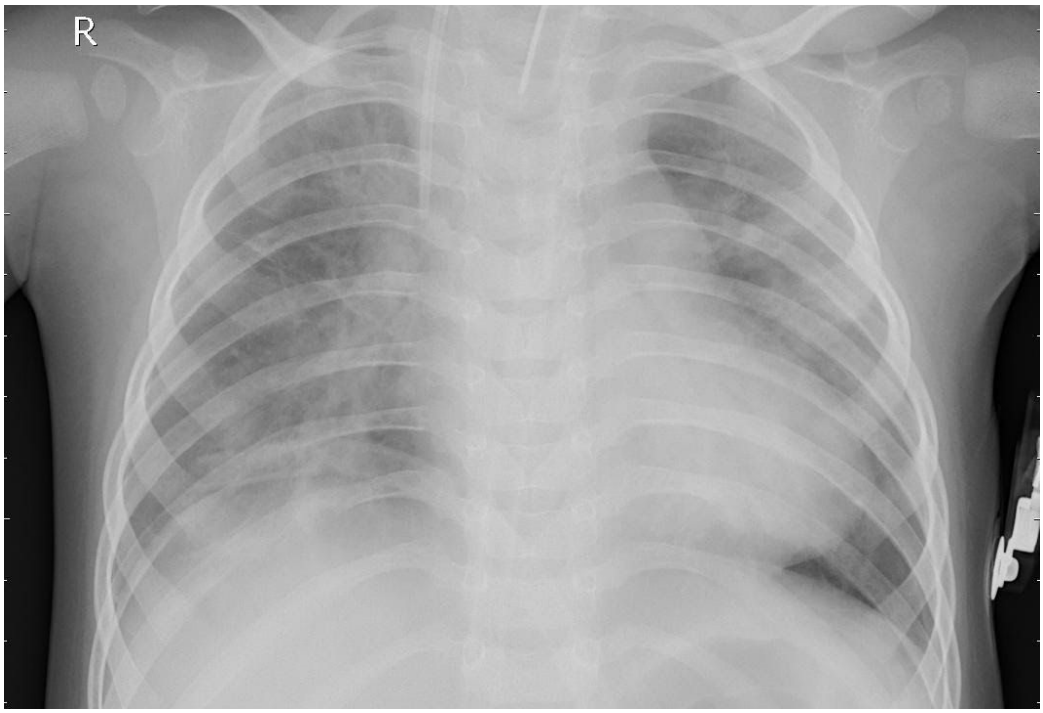


Fig 2.A.2.2(Pneumonia X-ray example)



2.2.2 Data Handling CNN

For use in experimentation with the Convolutional Neural Network the dataset was handled using the libraries Pandas and Keras. The image data was preprocessed using the keras preprocessing functionality implemented through the ImageDataGenerator class. Image data was scaled to 1/255 with a zoom range of 0.2 and extrapolated as grayscale values. The greyscale values generated by the keras functionality was then fit to a dataframe using pandas. The

usability of this dataset for a CNN algorithm was very high and allowed for a highly accurate classification model.

2.2.3 Data Handling RDF

Preprocessing of image data for the Random Decision Forest Classifier involves loading images using the keras ImageDataGenerator class, reading from directory using flow from directory with grayscale color mode. A loop is used to aggregate all of the image data into a list due to the way ImageDataGenerator process and generate the image data in mini-batches, which is not compatible with scikit-learn's RandomForestClassifier class which does not take mini-batches. Further preprocessing to make the image data compatible with RandomForestClassifier involves reshaping the list of features from (5216, 256, 256, 1), which represents number of samples, pixel width, pixel height, channel, into (5216, 65536) and translating the list of target label from a 2-Dimensional binary class membership list to a 1-Dimensional list of numbers that corresponds to a class label.

2.3 Experiment Description/ analysis

2.3.1 Experiment Overview

The experiment conducted for this report is an attempt to validate whether it is possible to detect and diagnose pneumonia from X-ray images of the human lungs. Our experiment tested whether using Supervised machine learning algorithms (CNN & RDF) trained on well screened and clean data can produce reasonable accuracy output of 85% or higher. The experimentation

was done by developing the two machine learning models in a python environment and training it before testing it against a test and validation set of data.

The experiment for the uses of this reports hypothesis are served well but when examining the dataset used we may find that our results are skewed or biased. Whether we find that the accuracy achieved by the used Machine Learning algorithms is acceptable or not we can not say for certain that these models are necessarily an acceptable replacement for a physician diagnosis. The dataset used for this experiment have been properly and thoroughly vetted for clear diagnosable images. If the dataset we used for this experiment had not been manicured so precisely it may have lead to far lower accuracy results.

2.4 Code Analysis

2.4.1 CNN Code

The Convolutional Neural Network algorithm was implemented using the Keras sequential model and compiled with the built in fit_generator() function. The layers of the CNN used by the sequential model were as follows: Convolutional, Convolutional, Batch Normalization, Max pooling, Dropout, Convolutional, Convolutional, Batch Normalization, Max Pooling, Dropout, Flatten, Dense, Batch Normalization, Dropout and a final Dense layer. The activation function added to this model for all layers was the Relu activation function. The Dropout rate used was 0.25 and the pooling layer were 2x2. Data augmentation and preprocessing was done using Keras and Pandas while data visualization was done using Matplotlib and the Keras History class.

2.4.2 RDF Code

The Python implementation of the Random Decision Forest algorithm uses the RandomForestClassifier class from the ensemble module of scikit-learn package with default hyperparameters and two sets of tuned hyperparameter values which were searched using RandomizedSearchCV and GridSearchCV classes from the model selection module of scikit-learn. Code for the hyperparameter tuning process was adapted from an article written by Will Koehrsen which source can be found in the references section at the end of the document. The following are the most significant hyperparameters considered in the tuning process of the Random Forest Classifier:

- Number of estimators - Number of trees in the forest,
- Max. depth - Maximum depth of each trees,
- Min. sample split - Minimum number of samples required to split a node,
- Min. leaf - Minimum number of samples to be at the leaf node,
- Max. features - Number of features to consider when looking for the best split,
- Bootstrap - Bagging/ bootstrap aggregation is true/ false; Determines whether replacement sample data is enabled/ disabled during random distribution of sample data for the tree creation.

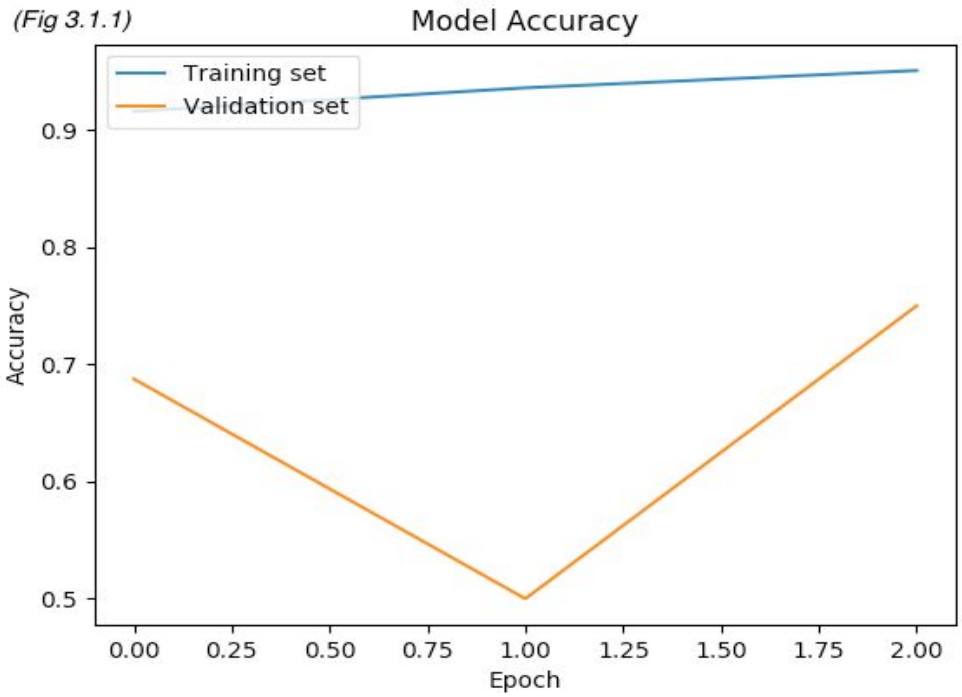
Training and testing images were loaded and preprocessed with keras ImageDataGenerator, where the RandomForestClassifier model is then made to fit the training data features and target labels; Prediction is then done on the testing features and the prediction results is compared with testing data target labels for accuracy measurements. Pandas, Seaborn and Matplotlib packages were used to plot the confusion matrix.

3.Results

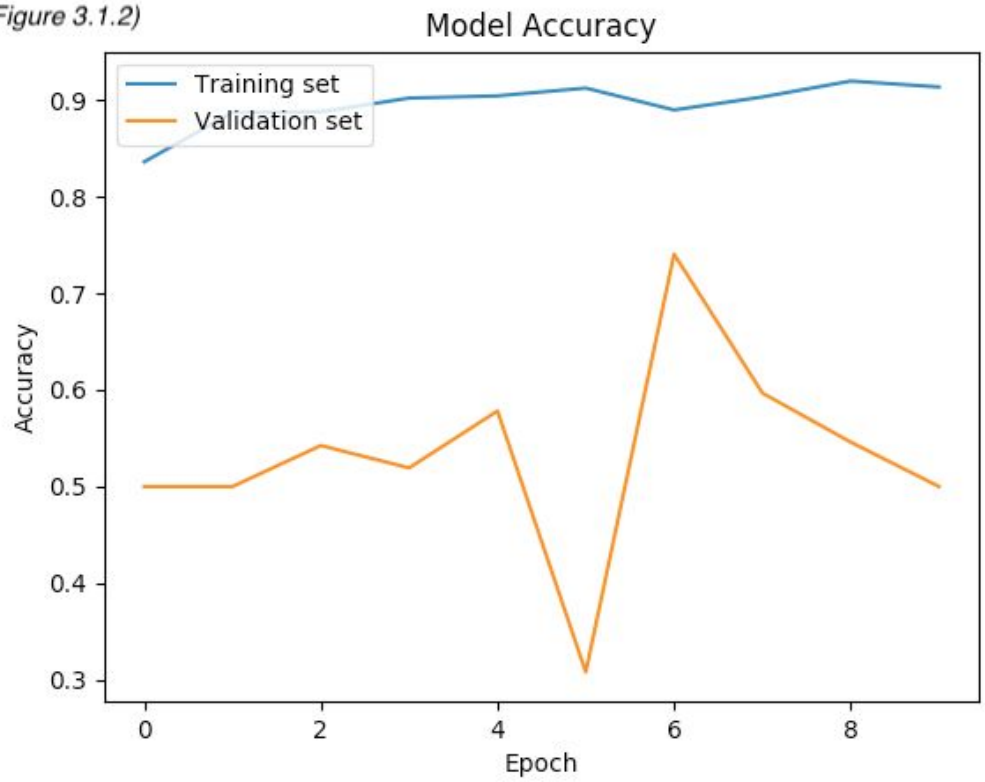
3.1 CNN Experimentation Results

For experimentation I ran the CNN classification algorithm with 3 epochs and 10 epochs and also altered the hyperparameters associated with the number of steps taken per epoch and the number of validation steps. Through trial and error I found that the best values for number of steps per epoch and steps per validation step was 163. When running the experiment with just 3 epochs (fig 3.1.1) the training accuracy was 92.3% and the testing accuracy was 84.15% while the validation accuracy ended at 72.68%. When running the experiment with 10 epochs (fig 3.1.2) the training accuracy was 92.75% and the testing accuracy was 86.20% with a validation set accuracy peaking at 72% accuracy and ending at 55% accuracy.

(Fig 3.1.1)



(Figure 3.1.2)



3.2 RDF Experimentation results

Figure 2.B.1.b.1 describes the hyperparameter values used in the three experiments. Results of the experiment is shown in Figure 2.B.1.b.2. Additionally, the confusion matrix of each experiment results are given in Figure 2.B.1.b.3, 2.B.1.b.4, and 2.B.1.b.5.

Fig 2.B.1.b.1(Hyperparameter Values)

	Default	Random Search Result	Grid Search Result
Bootstrap	TRUE	FALSE	FALSE
Number of Estimators	10	1000	300
Max. Features	auto*	auto*	auto*
Min. Sample Split	2	10	8
Min Samples Leaf	1	1	3
Max Depth	None	80	90

**Auto = square root of number of features*

Fig 2.B.1.b.2(Prediction Results)

	Default	Random Search Result	Grid Search Result
True Positive*	383	387	387
True Negative**	104	85	101
False Positive*	130	149	133
False Negative**	7	3	3
Accuracy	0.7804	0.7564	0.7821

**Positive = Pneumonia*

***Negative = Normal*

Fig 2.B.1.b.3(Confusion Matrix - Default Hyperparameter Set)

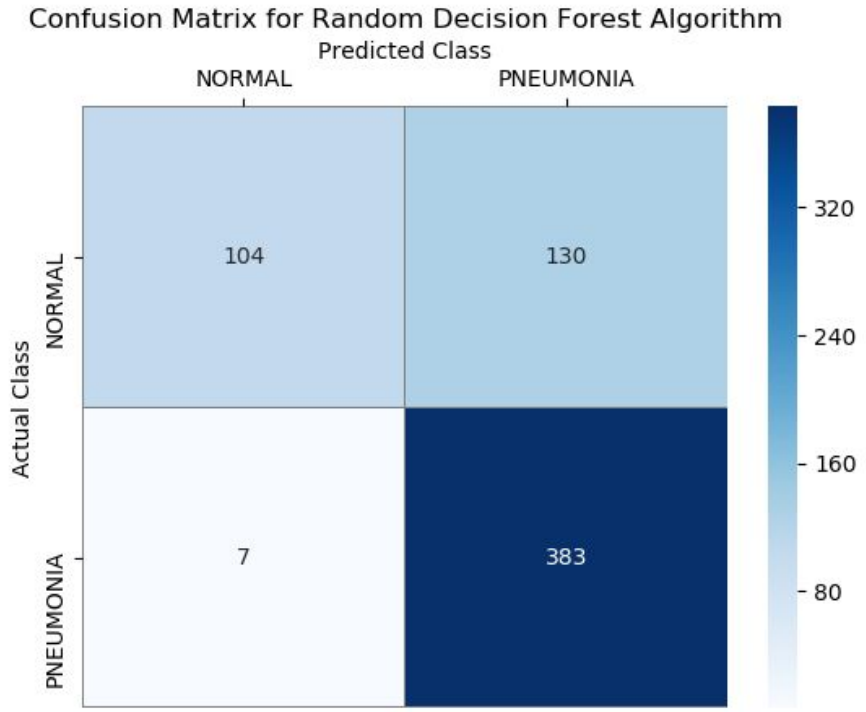


Fig 2.B.1.b.4(Confusion Matrix - Random Search Best Hyperparameter Set)

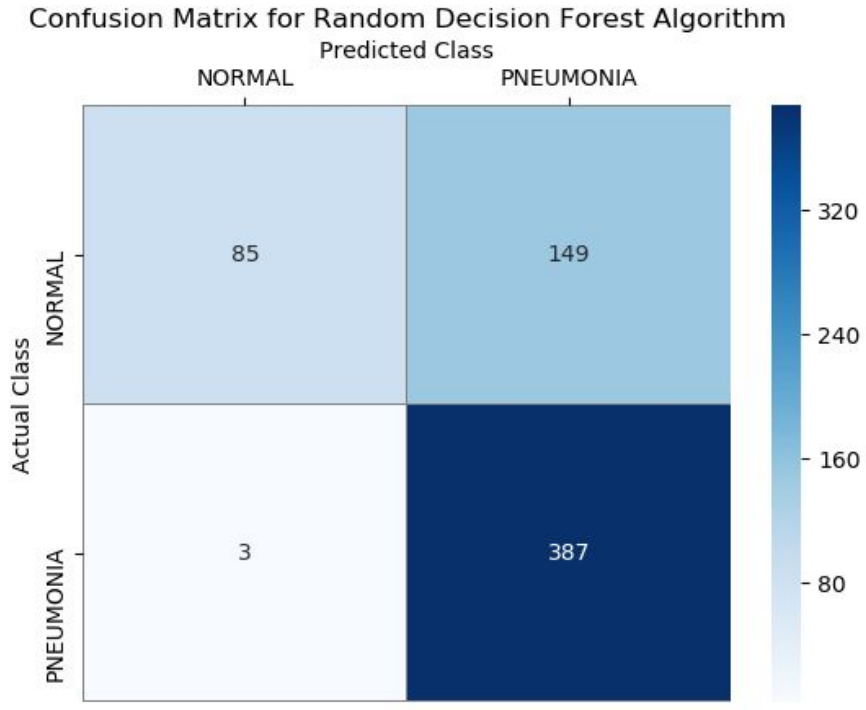
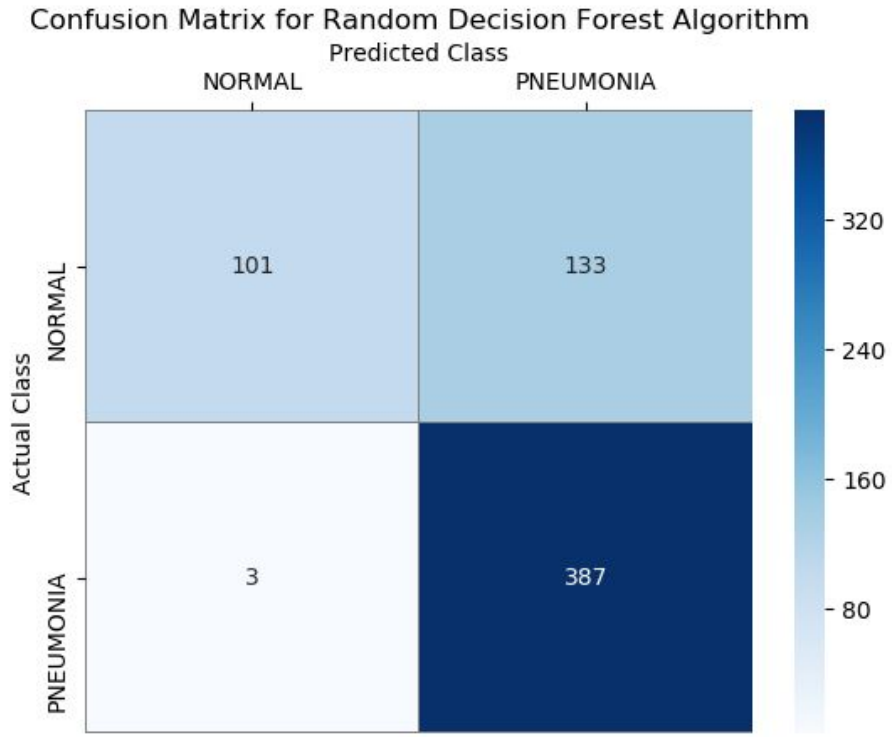


Fig 2.B.1.b.5(Confusion Matrix - Grid Search Best Hyperparameter Set)



4. Discussion

4.1 CNN results analysis:

The Convolutional Neural Network performed well on the given task and was able to meet the hypothetical accuracy of above 85% for predictions in the Training and Validation set but under some conditions the output of the test set were far lower than this threshold and it appears that there are signs of overfitting to the training data. In both subsegments of the experiment we can see that the number of epochs has very little difference of the training data accuracy but it does seem to have a pretty extreme effect on the accuracy for the validation set

with a definitive sign of oscillation appearing as soon as there are more than 3 epochs. Further tuning of hyperparameters such as the ordering and content of the hidden layer of this CNN may improve the results and curb the overfitting exhibited in this experiment.

4.2 RDF results analysis:

Accuracy of the Random Decision Forest Classifier in all of the results were less than the 85% hypothesized value. Using grid search best hyperparameter values increased accuracy by a small amount (≈0.001) when compared to the default hyperparameter values. Random hyperparameter search and grid search best hyperparameter values have four less false negatives than default hyperparameter values, but also more false positives. The accuracy of Random Decision Forest algorithm is lesser than Convoluted Neural Network as expected.

5. Conclusion

5.1 Summary:

Hyperparameter tuning only provides marginal improvement for the Random Decision Forest; Improvements to the hyperparameter tuning process can be made with a more thorough search of optimal hyperparameters using a larger range of values. To further improve the Random Decision Forest Classifier's prediction accuracy, a larger number of samples/ training dataset may be required. For the Convolutional neural network tweaking the hidden layers or adjusting the way the images are preprocessed and shuffled may improve its overfitting and oscillation issues.

5.2 Future Works:

future experiments using datasets that contains non-cleaned or lower quality X-ray images can be done to better simulate the efficacy of both algorithms on real world examples. It should also be noted that the given experiment is done using a dataset comprised only of binary classification of pneumonia or healthy X-ray images, a more robust experiment would include a third control set for X-ray images of unhealthy lungs not classified with a specific medical issue to prove that a Machine Learning model could diagnose Pneumonia not just classify whether it exists in an X-ray image.

6. References

Hubel, D H, and T N Wiesel. “Receptive fields and functional architecture of monkey striate cortex.” *The Journal of physiology* vol. 195,1 (1968): 215-43. doi:10.1113/jphysiol.1968.sp008455 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912/

Karpathy, Andrej. “Convolutional Neural Network for Visual Recognition.” *CS231n Convolutional Neural Networks for Visual Recognition*, 2018, cs231n.github.io/convolutional-networks/.

Koehrsen, W. (2019, Jan 9) Hyperparameter Tuning the Random Forest in Python. *Towards Data Science*.Retrieved from: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Yiu, T. (2019, June 12) Understanding Random Forest. *Towards Data Science*. Retrieved from: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Libraries used:
Keras Sequential: https://keras.io/models/sequential/
Matplotlib pyplot: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html
Pandas: https://pandas.pydata.org/
SciKitLearn: https://scikit-learn.org/stable/
Tensor flow: https://www.tensorflow.org/
Seaborn: https://seaborn.pydata.org/

Data set and experimentation concept provided via Kaggle:
https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Table of Contents